



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего
образования

"МИРЭА - Российский технологический университет"

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра математического обеспечения и стандартизации информационных
технологий (МОСИТ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №5_2
по дисциплине
«Структуры и алгоритмы обработки данных»

Тема. Работа с данными из файла

Выполнил студент группы ИКБО-42-23

Голев С.С.

Принял старший преподаватель

Муравьёва Е.А.

Москва 2024

ОГЛАВЛЕНИЕ

ЦЕЛЬ РАБОТЫ	3
1. УСЛОВИЯ ЗАДАЧ	4
2. ОТЧЁТ ПО ЗАДАНИЮ 1.....	5
2.1. Определение структуры	5
2.2. Код используемый в программе	6
3. ОТЧЁТ ПО ЗАДАНИЮ 2.....	9
3.1. Алгоритм линейного поиска	9
3.2. Код используемый в программе	9
3.3. Результаты тестирования	10
4. ОТЧЁТ ПО ЗАДАНИЮ 3.....	11
4.1. Алгоритм бинарного поиска	11
4.2. Код используемый в программе	11
4.2. Результаты тестирования	12
4. ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ	14

ЦЕЛЬ РАБОТЫ

Поучить практический опыт по применению алгоритмов поиска в таблицах данных.

1.УСЛОВИЯ ЗАДАЧ

Задание 1:

Создать двоичный файл из записей (структура записи определена вариантом). Поле ключа записи в задании варианта подчеркнуто. Заполнить файл данными, используя для поля ключа датчик случайных чисел. Ключи записей в файле уникальны.

Задание 2:

Поиск в файле с применением линейного поиска.

Задание 3:

Поиск записи в файле с применением дополнительной структуры данных, сформированной в оперативной памяти.

Бинарный поиск	Книга: ISBN – двенадцатизначное число, Автор, Название
----------------	---

2. ОТЧЁТ ПО ЗАДАНИЮ 1

2.1. Определение структуры

АТД s_book

{

Данные.

ISBN – двенадцатизначное число.

author – массив типа char с зарезервированными 50 ячейками.

name – массив типа char с зарезервированными 50 ячейками.

Операции.

1) Перевод текстового файла в бинарный файл;

//Предусловие. Текстовый файл

//Постусловие. Новый бинарный файл

convert_to_binary(const char* textFileName, const char* binaryFileName);

2) Перевод бинарного файла в текстовый файл;

//Предусловие. Бинарный файл,

//Постусловие. Новый текстовый файл

save_to_text(const char* binaryFileName, const char* textFileName);

3) Вывод содержимого бинарного файла на экран;

//Предусловие. Бинарный файл,

//Постусловие. Вывод содержимого бинарного файла на экран

print_all_records(const char* binaryFileName);

4) Удаление элемента по индексу в бинарном файле;

//Предусловие. Бинарный файл,

//Постусловие. Бинарный файл без i-ого элемента

access_record_by_index(const char* binaryFileName, int index);

}

2.2. Код используемый в программе

Опишем функции, используемые в программе для решения задач.

```
void convert_to_binary(const char* textFileName, const char* binaryFileName) {
    std::ifstream inputFile(textFileName);
    std::ofstream outputFile(binaryFileName, std::ios::out | std::ios::binary);

    if (!inputFile.is_open()) {
        std::cout << "Ошибка открытия текстового файла" << std::endl;
        return;
    }
    if (!outputFile.is_open()) {
        std::cout << "Ошибка создания двоичного файла" << std::endl;
        return;
    }

    s_book record;
    while (inputFile >> record.ISBN >> record.author >> record.name) {
        outputFile.write((char*)&record, sizeof(s_book));
    }

    inputFile.close();
    outputFile.close();

    std::cout << "Преобразование завершено" << std::endl;
}
```

Рисунок 2.1 – Функция для перевод текстового файла в бинарный файл

```

void save_to_text(const char* binaryFileName, const char* textFileName) {
    std::ifstream inputFile(binaryFileName, std::ios::binary);
    std::ofstream outputFile(textFileName);

    if (!inputFile.is_open()) {
        std::cout << "Ошибка открытия двоичного файла" << std::endl;
        return;
    }
    if (!outputFile.is_open()) {
        std::cout << "Ошибка создания текстового файла" << std::endl;
        return;
    }

    s_book record;
    while (inputFile.read(reinterpret_cast<char*>(&record), sizeof(s_book))) {
        outputFile << record.ISBN << std::endl << record.author << std::endl << record.name <<
std::endl;
    }

    inputFile.close();
    outputFile.close();

    std::cout << "Сохранение в текстовый файл завершено" << std::endl;
}

```

Рисунок 2.2 – Функция перевод бинарного файла в текстовый файл

```

void print_all_records(const char* binaryFileName) {
    std::ifstream inputFile(binaryFileName, std::ios::binary);
    if (!inputFile.is_open()) {
        std::cout << "Ошибка открытия двоичного файла" << std::endl;
        return;
    }

    s_book record;
    while (inputFile.read(reinterpret_cast<char*>(&record), sizeof(s_book))) {
        std::cout << "ISBN: " << record.ISBN << ",\tАвтор: " << record.author << ",\tНазвание: "
<< record.name << std::endl;
    }

    inputFile.close();
}

```

Рисунок 2.3 – Функция для вывода содержимого бинарного файла на экран

```

void access_record_by_index(const char* binaryFileName, int index) {
    std::ifstream inputFile(binaryFileName, std::ios::binary);
    if (!inputFile.is_open()) {
        std::cout << "Ошибка открытия двоичного файла" << std::endl;
        return;
    }

    s_book record;
    inputFile.seekg(index * sizeof(s_book));

    if (inputFile.read(reinterpret_cast<char*>(&record), sizeof(s_book))) {
        std::cout << "ISBN: " << record.ISBN << ",\tАвтор: " << record.author << ",\tНазвание: "
        << record.name << std::endl;
    }
    else {
        std::cout << "Запись с указанным номером не найдена" << std::endl;
    }

    inputFile.close();
}

```

Рисунок 2.4 – Функция для удаления элемента по индексу в бинарном файле

3. ОТЧЁТ ПО ЗАДАНИЮ 2

3.1. Алгоритм линейного поиска

Алгоритм линейного поиска представляет собой обычный проход по всем элементам файла, пока не будет найден нужный элемент.

3.2. Код используемый в программе

Опишем функции, используемые в программе для решения задач.

```
void lin_search(const char* binaryFileName, char ISBN[13])
{
    std::ifstream inputFile(binaryFileName, std::ios::binary);
    if (!inputFile.is_open()) {
        std::cout << "Ошибка открытия двоичного файла" << std::endl;
        return;
    }

    s_book record;
    while (inputFile.read(reinterpret_cast<char*>(&record), sizeof(s_book))) {
        bool flag = true;
        for (int i = 0; i < 13; i++)
        {
            if (ISBN[i] != record.ISBN[i])
                flag = false;
        }
        if (flag)
            std::cout << "ISBN: " << record.ISBN << ",\tАвтор: " << record.author << ",\tНазвание: " << record.name << std::endl;
    }

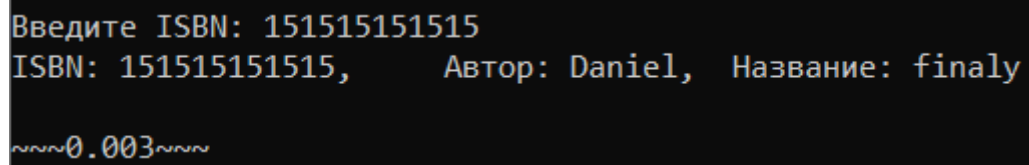
    inputFile.close();
}
```

Рисунок 3.1 – Функция линейного поиска

Данная функция на вход получает имя бинарного файла и ключ, после выполнения выводится элементы файла, соответствующие индексу ключа.

3.3. Результаты тестирования

Измерим время для поиска по ключу, в файле с 100 элементами (нужный ключ 84 элемент)

A screenshot of a terminal window with a black background and yellow text. The text shows a prompt 'Введите ISBN: 151515151515', followed by the input 'ISBN: 1515151515,' and the output 'Автор: Daniel, Название: finaly'. The final line shows the execution time '~~~0.003~~~'.

```
Введите ISBN: 151515151515
ISBN: 1515151515,      Автор: Daniel,  Название: finaly
~~~0.003~~~
```

Рисунок 3.2 – Тестирование функции, реализующей линейный поиск
Линейный поиск файла занял 0,003 секунды.

4. ОТЧЁТ ПО ЗАДАНИЮ 3

4.1. Алгоритм бинарного поиска

Бинарный поиск — тип поискового алгоритма, который последовательно делит пополам заранее отсортированный массив данных, чтобы обнаружить нужный элемент.

4.2. Код используемый в программе

Опишем функции, используемые в программе для решения задач.

```
void bin_search(const char* binaryFileName, char ISBN[13])
{
    std::ifstream inputFile(binaryFileName, std::ios::binary);
    if (!inputFile.is_open()) {
        std::cout << "Ошибка открытия двоичного файла" << std::endl;
        return;
    }

    s_book record;
    std::vector<std::string> vec;

    while (inputFile.read(reinterpret_cast<char*>(&record), sizeof(s_book))) {
        char el[13];
        for (int i = 0; i < 13; i++)
        {
            el[i] = record.ISBN[i];
        }
        vec.push_back(el);
    }
    inputFile.close();

    clock_t start = clock();
    std::string search = ISBN;
```

Рисунок 4.1 – Функция, реализующая бинарный поиск(часть 1/2)

```

int index;
int i = vec.size() / 2;
int delta = vec.size() / 2;
while (true)
{
    if (vec.at(i) == search)
    {
        index = i;
        break;
    }
    if (vec.at(i) < search)
    {
        i += delta / 2;
        if (delta != 1)
            delta = delta / 2;
    }
    if (vec.at(i) > search)
    {
        i -= delta / 2;
        if (delta != 1)
            delta = delta / 2;
    }
}

clock_t end = clock();
double seconds = (double)(end - start) / CLOCKS_PER_SEC;

std::ifstream outputFile(binaryFileName, std::ios::binary);

outputFile.seekg(index * sizeof(s_book));
if (outputFile.read(reinterpret_cast<char*>(&record), sizeof(s_book)))
    std::cout << "ISBN: " << record.ISBN << ",\tАвтор: " << record.author << ",\tНазвание: "
<< record.name << std::endl;
outputFile.close();
std::cout << std::endl << "~~~" << seconds << "~~~" << std::endl;
}

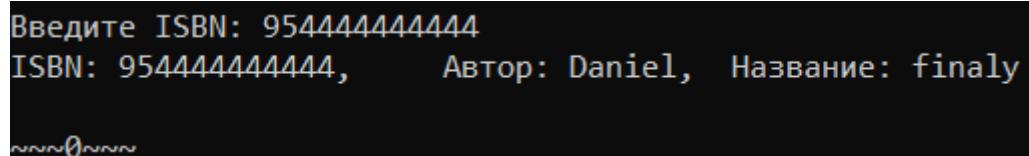
```

Рисунок 4.2 – Функция, реализующая бинарный поиск(часть 2/2)

Данная функция на вход получает имя бинарного файла и ключ, после выполнения выводится элементы файла, соответствующие индексу ключа.

4.2. Результаты тестирования

Представим примеры работы программы.



```
Введите ISBN: 954444444444
ISBN: 954444444444,    Автор: Daniel,  Название: finaly
~::~~
```

Рисунок 4.3 – Тестирование функции, реализующей бинарный поиск

Данная функция отработала на том же файле что и линейный поиск, но она справилась за куда меньшее время. Что говорит об эффективности этого алгоритма

4.ИСПОЛЬЗУЕМЫЕ ИСТОЧНИКИ

1. Лекции по Структуры и алгоритмы обработки данных / Рысин М. Л. Москва, МИРЭА — Российский технологический университет.
2. Материалы по дисциплине Структуры и алгоритмы обработки данных / Скворцова Л. А. Москва, МИРЭА — Российский технологический университет.